Construction of Visual Classifier by Edge Crossing Minimization (The evolution of optimization models and algorithms)

AUTHOR(S):

Haraguchi, Kazuya; Hong, Seok-Hee; Nagamochi, Hiroshi

数理解析研究所講究録
第1726巻 2011年 73-83

Construction of Visual Classifier by Edge Crossing Minimization

Kazuya Haraguchi* Seok-Hee Hong† Hiroshi Nagamochi‡

Abstract

We consider a machine learning problem called *classification*. In this problem, we are given a training set of examples, where each example is represented by a nominal-valued vector and belongs to one of the pre-defined classes. We are asked to construct a *classifier* that predicts the classes of future examples with high accuracy. We have worked on developing a new *visual classifier* so far which can provide us with insights into data by its drawing beyond a mathematical function. In this article, we review the visual classifier which was studied in our previous works, in order to realize future directions of our research.

1 Introduction

In this paper, we consider a machine learning problem called *classification*, which is a fundamental but a significant issue from classical statistics to modern fields on learning theory [7].

For a positive integer i, we denote $[i] = \{1, 2, \ldots, i\}$. In this problem, we are given a *training set* X of *examples*. An example $x \in X$ is specified by entries on n *attributes* (and thus is represented by an n dimensional vector), and belongs to one of the pre-defined *classes*. The set of classes is denoted by C. For any $j \in [n]$, let D_j denote the domain of attribute j. In this paper, we assume that each D_j is a finite set of discrete elements. We call the product $\mathcal{X} = D_1 \times \cdots \times D_n$ the *example space*. The aim of the problem is to construct a *classifier*, a function from the example space $\mathcal{X}$ to the class set C, that predicts the class of a "future" example $y \in \mathcal{X}$ with high accuracy, where y is possibly unseen, i.e., $y \notin X$. In particular, when $|C| = 2$ (resp., $|C| > 2$), the problem is called *binary classification* (resp., *multiclass classification*).

In our previous research, we have worked on developing a new *visual classifier*, which can provide us with insights into data by its drawing, inspired by many successes of *visualization* and *graph drawing*. Given a good visualization of abstract data, one may expect that a hidden structure has been revealed. Our main hypothesis is that good visualization (e.g., visual objects with low visual complexity) itself can discover essential or hidden structure of data without relying on any data analysis techniques, which can lead to a novel learning technique.

Our strategy for constructing a visual classifier consists of the following procedure.

1: Extract a graph structure that contains the features of X well. (The X is usually given as a spreadsheet.)

2: Compute a "good" visualization of the graph.

3: Determine the mechanism to utilize the well-drawn graph as a classifier.

*Faculty of Science and Engineering, Ishinomaki Senshu University, Japan (kazuyah@isenshu-u.ac.jp)
†School of Information Technologies, University of Sydney, Australia (shhong@it.usyd.edu.au)
‡Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University (nag@amp.i.kyoto-u.ac.jp)

Table 1: An instance of training set $X = X^+ \cup X^-$

		Att. 1 (headache)	Att. 2 (temperature)	Att. 3 (blood pressure)
X^+ (malignant)	x_1	yes	high	high
	x_2	yes	med	med
	x_3	yes	high	high
	x_4	no	high	med
X^- (benign)	x_5	no	med	high
	x_6	yes	high	med
	x_7	no	med	low

After designing some prototypes [10, 11], we proposed our binary visual classifier [9, 12, 13] and extended it for multiclass classification [14]. Our visual classifier is based on what we call *entry-example graph* (*EX-graph*), which is constructed from a set of *decision tables* and the training set X.

In this article, we review our previous work on visual classifier so far and discuss future work in order to realize directions of our research. The article is organized as follows: In Section 2, we explain how to extract EX-graph from X and how to construct binary visual classifier from EX-graph. Then in Section 3, we describe how to extend it for multiclass classification by using *edge crossing minimization technique*. We then discuss our future work in Section 4. Note that the discussions in Sections 2 and 3 are mainly from [9] and [14], respectively.

2 Visual Classifier for Binary Classification

2.1 Overview

We focus on binary classification in this section. Let us take $C = \{+, -\}$ where $(+)$ (resp., $(-)$) represents the *positive* (resp., *negative*) class. Table 1 shows an instance of training set $X = X^+ \cup X^-$ for binary classification, where X^+ (resp., X^-) denotes the set of positive (resp., negative) examples in X. In this X, each example corresponds to a patient of some disease. The three attributes represent headache, temperature, and blood pressure respectively, and their domains are defined as $D_1 = \{\texttt{yes}, \texttt{no}\}$, $D_2 = \{\texttt{high}, \texttt{med}\}$, and $D_3 = \{\texttt{high}, \texttt{med}, \texttt{low}\}$.

Our visual classifier is based on EX-graph. In order to explain EX-graph, we need to introduce decision table. Formally, a decision table $T = (A, \ell)$ is such a classifier that is defined by a subset $A = \{j_1, \ldots, j_q\} \subseteq [n]$ of n attributes and a *label function* $\ell : D_{j_1} \times \cdots \times D_{j_q} \to \{+, -\}$. That is, the label function ℓ assigns either $(+)$ or $(-)$ to each entry of A. Table 2 shows three decision tables for the training set in Table 1. For a future example, a decision table infers its class as the label of the matched entry. For example, $(\texttt{no}, \texttt{high}, \texttt{high})$ is classified into $(+)$ by T_1, $(-)$ by T_2 and $(+)$ by T_3.

In the sequel, we assume that a set $\mathcal{T} = \{T_1, T_2, \ldots, T_K\}$ of decision tables is given. We do not focus on how to generate it here. This issue will be addressed in our future papers. Let $K = |\mathcal{T}|$ denote the number of decision tables in $\mathcal{T}$. We denote the k-th decision table by $T_k = (A_k, \ell_k)$. Following the procedure described in Section 1, we summarize how to construct binary visual classifier as follows.

Extraction of EX-graph. To construct a visual classifier, we first extract EX-graph for training set X and decision table set $\mathcal{T}$. Denoted by $G = (X, \mathcal{D}, E)$, EX-graph is a bipartite graph, where one node set corresponds to the examples in X, the other node set corresponds to the entries $\mathcal{D}$

Table 2: Decision tables $T_1 = (A_1, \ell_1)$, $T_2 = (A_2, \ell_2)$ and $T_3 = (A_3, \ell_3)$ with attribute sets $A_1 = \{1, 2\}$, $A_2 = \{1, 3\}$ and $A_3 = \{3\}$

$T_1 = (A_1, \ell_1)$

$v \in D_1 \times D_2$	$\ell_1(v)$
yes, high	+
yes, med	+
no, high	+
no, med	−

$T_2 = (A_2, \ell_2)$

$v \in D_1 \times D_3$	$\ell_2(v)$
yes, high	+
yes, med	+
yes, low	+
no, high	−
no, med	+
no, low	−

$T_3 = (A_3, \ell_3)$

$v \in D_3$	$\ell_3(v)$
high	+
med	+
low	−

of the decision tables in $\mathcal{T}$, and E denotes the edge set. We denote by $\mathcal{D}_k$ the set of entries of decision table $T_k \in \mathcal{T}$, and then we have $\mathcal{D} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_K$. An example node and an entry node are joined by edge if and only if the example matches the entry. Figure 1(a) shows the EX-graph for X in Table 1 and $\mathcal{T} = \{T_1, T_2, T_3\}$ in Table 2, where the entry values are abbreviated, e.g., "nh" of T_1 stands for "no, high."

Good visualization of EX-graph. We employ *two-layered drawing* to draw an EX-graph, where the two node sets are laid on parallel layers. A *drawing* (σ, π) of the EX-graph is defined as the ordering (i.e., bijection) $\sigma : X \to [|X|]$ on the example nodes and the ordering $\pi : \mathcal{D} \to [|\mathcal{D}|]$ on the entry nodes. For π, we restrict ourselves to the permutations where the entries from the same decision table are arranged consecutively. Then π is decomposed into $\pi_1, \ldots, \pi_K$, where $\pi_k : \mathcal{D}_k \to [|\mathcal{D}_k|]$ for each $k \in [K]$.

As mentioned in the literature (e.g., [17, 24]), crossing minimization may improve readability of EX-graph. Then we try to minimize the number of edge crossings in order to obtain a good visualization of EX-graph. Let us describe the key idea as follows: Observe that some entry nodes may be connected to positive example nodes more than negative ones, or vice versa. We decide the ordering π on the entry nodes by *positivity*, i.e., the bias of the classes in the matching examples. For example, see the upper layer of Figure 1(b). Among the entry nodes of T_1, "nm" is laid on the left since it is not connected to any positive example nodes, while "ym" and "nh" are laid on the right since they are connected only to positive example nodes. For the fixed π, we perform *one-sided edge crossing minimization* (1CM) to determine the ordering σ on the example nodes. We expect 1CM to reduce the "conflict" of the two orderings, and thus to separate the positive and negative example nodes, as shown in the lower layer of Figure 1(b). Since 1CM is NP-hard [6], we need to employ an approximation algorithm to solve it.

Formally, we denote by $\rho_k : \mathcal{D}_k \to [-1, 1]$ $(\forall k \in [K])$ the above mentioned positivities of entry nodes of decision table T_k. For an entry $v \in \mathcal{D}_k$, we denote by $\mu^+(v)$ (resp., $\mu^-(v)$) the number of positive (resp., negative) examples in X that match v. We utilize the following definition of $\rho_k(v)$ which is called *precision* in the literature [18];

$$\rho_k(v) = \begin{cases} 0 & \text{if } \mu^+(v) = \mu^-(v) = 0, \\ \frac{2\mu^+(v)}{\mu^+(v)+\mu^-(v)} - 1 & \text{otherwise.} \end{cases} \tag{1}$$

We then decide the ordering π_k as the non-decreasing order of ρ_k, i.e., we set $\pi_k(v_1) = 1$, $\pi_k(v_2) = 2$, $\ldots$, $\pi_k(v_{\mathcal{D}_k}) = |\mathcal{D}_k|$ so that $\rho_k(v_1) \leq \rho_k(v_2) \leq \cdots \leq \rho_k(v_{\mathcal{D}_k})$.

To solve 1CM for fixed $\pi_1, \ldots, \pi_K$, there are several approximation algorithms available. For example, Eades and Wormald [6] proposed a *median method*, which produces a 3-approximate solution. The *barycenter heuristic* by Sugiyama et al. [26] is an $O(\sqrt{m})$-approximation algorithm

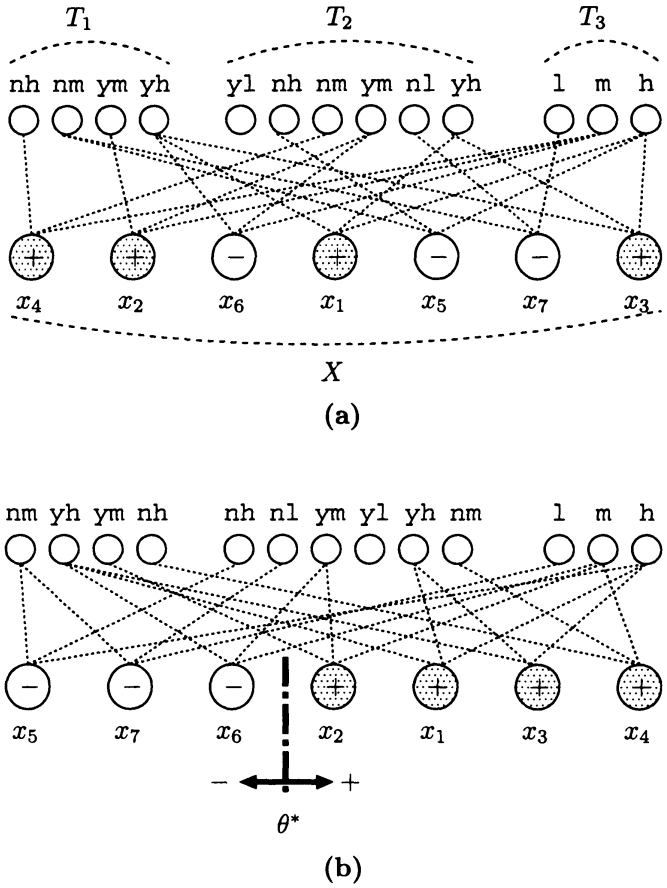

Figure 1: (a) A drawing of the EX-graph. (b) The drawing obtained by 1CM and the classifying threshold for SS model.

(where m denotes the number of nodes) [6]. Currently, the best known approximation algorithm is given by Nagamochi [22] that delivers a drawing with a 1.4664 factor approximation, based on a random key method. Among these, we utilize barycenter heuristic [26], which has been recognized as an effective approximation algorithm in practice, e.g., [19]. Barycenter heuristic permutes the examples x's in X in the non-decreasing order of barycenter $\beta(x)$, which is defined as follows;

$$\beta(x) = \frac{1}{K} \sum_{k \in [K]} \rho_k(x|_{A_k}) \tag{2}$$

Use of EX-graph as a classifier. We developed three visual classifier models using EX-graph, *string split* (*SS*) [9], *neighborhood majority* (*NM*) [12] and *example poset* (*XP*) [13]. The SS splits the example string which is computed by barycenter heuristic into positive and negative parts by a suitably chosen threshold (see θ^* in Figure 1(b)), and classifies a future example according to which side of the threshold it falls on. The NM defines the neighborhood of an example as an interval of the example string, by utilizing a partial order on X. The partial order is defined by the component-wise order of the vectors whose components are positivities of the adjacent entry nodes. Then NM classifies a future example into the majority class in its neighborhood. As an extension of NM, XP extracts a partially ordered set of examples and partitions it into positive and negative subsets. A future example is classified according to which subset it should belong to.

Among the three models, SS is the most successful one in terms of prediction accuracy. In SS, we compute the barycenter $\beta(y)$ in (2) of a future example $y \in \mathcal{X}$ and classify it into positive (resp., negative) if $\beta(y) > \theta^*$ (resp., $\beta(y) \leq \theta^*$), where θ^* is determined from the example string of X as follows:

$$\theta^* = \arg \min_{\theta \in [-1,1]} \frac{1}{|X|} \left(|\{x^+ \in X^+ \mid \beta(x^+) \leq \theta\}| + |\{x^- \in X^- \mid \beta(x^-) > \theta\}| \right). \tag{3}$$

That is, the above θ^* minimizes the training error. The paper [9] discusses the computational complexity for constructing SS model, and we omit the detail here.

2.2 Experimental Results

To observe prediction accuracy, we compare SS with C4.5 [25] and LibSVM [2] in terms of error rate. We take 14 data sets from UCI Repository [1] for benchmark instances. Table 3 shows the names of the taken data sets, along with the error rates of the three classifiers. These data sets have numerical and/or categorical attributes, and in order to treat them in our formulation, we transform example vectors into binary vectors by the method proposed in [15]. The error rates are estimated by 10-*fold cross validation* [27] except IONO, MONKS-1, MONKS-2 and MONKS-3 data sets which have their own test sets. Let us mention the used parameter values for the three classifiers as follows.

SS: To generate set $\mathcal{T}$ of decision tables, we use DECISIONTABLE package of Weka [20, 28]. This package generates a "good" attribute set by *local search*. By choosing the initial solution at random, we can generate different attribute sets, which results in different decision tables. We set all parameters to the default values except initial solution. We fix $K = |\mathcal{T}| = 30$, and take the minimum of 10 error rates over 10 different $\mathcal{T}$'s for evaluation.

C4.5: We test 8 combinations of parameter values: we set `confidence rate` to 1%, 25% (default), 50% or 99%, `binary split` option to `true` or `false` (default), and the other parameters to the default values. We take the minimum of 8 error rates for evaluation.

LibSVM: We test 32 combinations of parameter values: we use binary C-SVM and RBF kernel, and set $C = 0.5$, 1.0 (default), 2.0 or 4.0, $\gamma = 0.0$ (default), 0.5, 1.0 or 2.0, `normalization` option to `on` or `off` (default), and the other parameters to the default values. We take the minimum of 32 error rates for evaluation.

As shown in Table 3, our visual classifier based on SS model achieves competitive error rates with C4.5 and LibSVM, standard classifiers in the literature, although parameter values are hardly tuned for SS but are tuned for C4.5 and LibSVM. In this table, bold face represents the smallest error rate for each data set.

For computation time, C4.5 is the fastest among all, SS is the second, and LibSVM is the worst in general. (We omit the details due to space limitation.) For SS, more than 95% of computation time is devoted to construction of $\mathcal{T}$, for which we use Weka. This encourages us to develop an effective and efficient algorithm to construct $\mathcal{T}$.

3 Visual Classifier for Multiclass Classification

3.1 Overview

Let us refer to binary visual classifier introduced in Section 2 as 2-*SS*. The 2-SS can be extended to N-SS ($N > 2$) naturally. In 2-SS, a future example $y \in \mathcal{X}$ is classified according to its barycenter $\beta(y)$ in (2) (which is the average of positivities in (1) of the matched entries) is larger than the

Table 3: Error rates (%) of SS, C4.5 and LibSVM for 2 class data sets

Data	SS	C4.5	LibSVM
BCW	4.00	5.00	**3.43**
CHESS	1.78	**0.44**	0.59
HABERMAN	27.08	**26.44**	26.45
HEART	15.19	18.52	**14.07**
HEPATITIS	20.13	19.96	**19.42**
IONO	5.30	4.64	**3.31**
MONKS-1	**0.00**	**0.00**	8.56
MONKS-2	22.69	29.63	**18.75**
MONKS-3	3.47	**0.00**	2.31
MUSHROOM	**0.00**	**0.00**	**0.00**
PIMA	23.83	26.43	**23.57**
TICTACTOE	17.96	5.85	**2.71**
VOTING	4.58	4.36	**3.67**
WDBC	**4.56**	5.62	**4.56**

threshold θ^* in (3) or not. This is equivalent to classifying y by *similarity function* as follows; we define the similarity function with positive (resp., negative) class as $\beta(y) - \theta^*$ (resp., $\theta^* - \beta(y)$), and classify y into the class having the largest similarity. This idea is easily extended to multiclass cases by determining similarity functions for all classes.

Furthermore, we can use some general frameworks to extend any binary classifier to multiclass one. One may find the following three methods in the literature: *one-to-all* [23], *one-to-one* [16] and *error correcting output codes* [5]. However, these frameworks hardly take into account the structural relationships of classes, although it must be smarter to decompose the entire problem into subproblems for fewer classes in some application domains. For example, Figure 2 shows the hierarchical structure of classes in GLASS data set from UCI Repository [1].

There are some studies that attempt to extract hierarchical structure of classes, which we call a *class tree*. In a class tree, there are N leaves, and each leaf corresponds to one of the N classes. Each inner node corresponds to a *meta-class*, representing the set of its descendant classes (i.e., leaves). Let $N' \leq N$ denote the number of children of an inner node. For this inner node, N'-class classifier is constructed, where each child constitutes one class. Starting from the root, a future example is passed to one of the N' children, which is decided by the N'-class classifier of the current node. This procedure is repeated until the example reaches a leaf. Finally, the example is classified into the class of the reached leaf. Most of the previous works concentrate on extracting a binary tree as the class tree to decompose an N-class problem into binary subproblems (e.g., [3, 21]).

Note that one should extract a nice class tree and decompose the given N-class problem into easier subproblems. In the context of class tree, N-SS can be regarded as a star (i.e., a tree consists only of the root and the N leaves), where no decomposition is made, in the sense that N classes are treated homogeneously in classifier construction. However, N-SS does not seem to work well on such data sets that have structural relationships between classes. On the other hand, binary tree based approaches do not always work well because binary tree is not the universal structure of classes.

Then in [14], we proposed a new multiclass visual classifier, named SS-TREE, that can extract *any* tree as a class tree. To extract a class tree, we employ edge crossing minimization on two-layered drawing of EX-graph again, but in this case, we contract example nodes from the same class into one node. We can control the structure of the resulting class tree by tuning the newly introduced parameter. In the extracted class tree, we use N'-SS as the classifier for an inner node

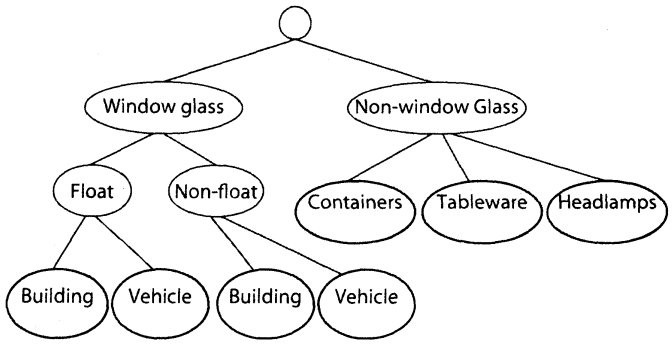

Figure 2: Hierarchical structure of classes in GLASS data set from UCI Repository [1]

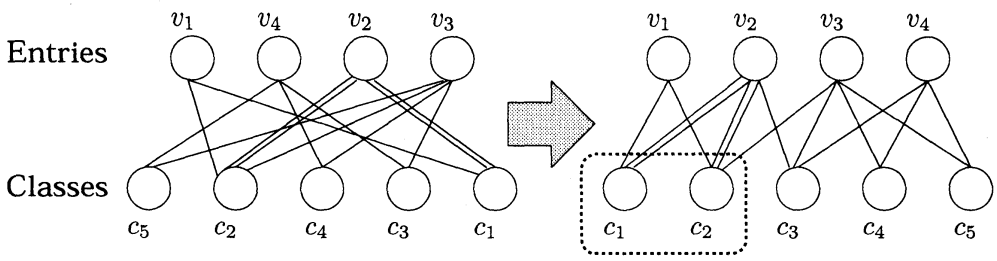

Figure 3: Two-sided edge crossing minimization (2CM) on EC-graphs

with N' children.

Algorithm to construct SS-tree. Let us denote by $C = \{c_1, c_2, \ldots, c_N\}$ the set of N classes. To construct a class tree from the given N-class training set X, we compute a laminar family $\mathcal{C} \subseteq 2^C$ of subsets of the class set C, and utilize $\mathcal{C}$ as the class tree. We include a subset $S \subseteq C$ in $\mathcal{C}$ if our criteria say that S should be treated as a meta-class in a binary (or at least fewer-class) subproblem. For this, we test if the examples from S and the examples from $\bar{S} = C \setminus S$ can be separated "effectively" by 2-SS.

To test the separability, we introduce *entry-class graph* (EC-graph), $\hat{G} = (C, \mathcal{D}, \hat{E})$, which is obtained from EX-graph $G = (X, \mathcal{D}, E)$ by contracting examples from the same class into one node $c_j \in C$. We define a layout of EC-graph by $(\hat{\sigma}, \pi_1, \ldots, \pi_K)$, where $\hat{\sigma} : C \to [N]$ is an ordering on the class set C and $\pi_k : \mathcal{D}_k \to [|\mathcal{D}_k|]$ $(k \in [K])$ is an ordering on the entry set $\mathcal{D}_k$.

Recall that 2-SS is obtained as a result of performing 1CM on EX-graph. We consider *two-sided edge crossing minimization* (2CM) on EC-graph that asks to compute the layout $(\hat{\sigma}, \pi_1, \ldots, \pi_K)$ to minimize the edge crossings. See Figure 3. In this simple example, we assume $N = 5$ and focus on one decision table with 4 entries. However, the discussion can be generalized easily.

Assume that 2CM has been performed on given EC-graph (see the right drawing of Figure 3). For convenience, let $\hat{\sigma}(c_1) = 1, \ldots \hat{\sigma}(c_N) = N$. In the figure, we can expect that the examples from $S = \{c_1, c_2\}$ and those from $\bar{S} = \{c_3, c_4, c_5\}$ can be separated by an appropriate construction of 2-SS. This expectation comes from the observation that there are few crossings between edges from S and $\bar{S}$, which may suggest the separability between S and $\bar{S}$, and that there are more edge crossings in inside of S, which may suggest that the examples of the classes in S match the similar entries (and thus take close values for barycenter).

Now let us formalize our criteria to decide whether $S \subseteq C$ should be included as a node in the

class tree. For $j \in [N]$ and $t \in [N-j] \cup \{0\}$, let $S_{j,t} = \{c_j, c_{j+1}, \ldots, c_{j+t}\}$ denote a consecutive subset of C. We define $\chi(j,t)$ to be the number of crossings between edges from $S_{j,t}$ and $\bar{S}_{j,t}$, and define $\eta(j,t)$ as follows;

$$\eta(j,t) = \begin{cases} 0 & \text{if } t = 0, \\ \max_{[j',t'] \subsetneq [j,t]} \frac{\chi(j,t)}{\chi(j',t')} & \text{otherwise.} \end{cases} \tag{4}$$

One can see that, if $\eta(j,t)$ is small, then the crossings between edges from $S_{j,t}$ and those from its outside, i.e., $\bar{S}_{j,t}$, are relatively fewer than the edge crossings inside $S_{j,t}$.

Our algorithm to construct a class tree is described in Algorithm 1. Whether $\eta(j,t)$ is "small" or not is decided by a positive parameter $\delta > 0$. In line 1, since the 2CM problem is NP-hard [8], we employ iterative application of barycenter heuristic in the experiments of the next subsection, i.e., repeat fixing one side and permuting the other side by barycenter heuristic until no change is made on both sides. In line 2, we can compute all $\eta(j,t)$'s efficiently by dynamic programming. However, the details are omitted due to space limitation.

Algorithm 1 CONSTRUCT-CLASS-TREE

1: Compute the layout $(\hat{\sigma}, \pi_1, \ldots, \pi_K)$ of EC-graph by 2CM.
2: For each $j \in [N]$ and $t \in [N-j] \cup \{0\}$, compute $\eta(j,t)$ by (4).
3: $\mathcal{I} \leftarrow \{S_{j,t} \subseteq C \mid j \in [N],\ t \in [N-j] \cup \{0\},\ \eta(j,t) < \delta\}$.
We denote $\mathcal{I} = \{S_{j_1,t_1}, \ldots, S_{j_b,t_b}\}$, where $\eta(j_a,t_a) \leq \eta(j_{a+1}, j_{a+1})$ $(\forall a \in [b-1])$.
4: $\mathcal{C} \leftarrow \emptyset$.
5: **for** $a \leftarrow 1, 2, \ldots, b$ **do**
6: **if** $\mathcal{C} \cup \{S_{j_a,t_a}\}$ is laminar **then**
7: $\mathcal{C} \leftarrow \mathcal{C} \cup \{S_{j_a,t_a}\}$.
8: **end if**
9: **end for**
10: Output $\mathcal{C}$.

SS-TREE is the visual classifier consisting of the class tree $\mathcal{C}$ and the N'-SS's for the inner nodes. Let us emphasize that our class tree is constructed based on edge crossing minimization on EC-graph. We describe some details as follows:

- The output $\mathcal{C}$ exactly includes singletons $S_{1,0} = \{c_1\}, \ldots, S_{N,0} = \{c_N\}$ for any $\delta > 0$ since $\eta(j,0) = 0 < \delta$ by (4). These N singletons serve as the leaves of the class tree.
- The parameter δ eventually controls the structure of the output class tree. Intuitively, if $\delta \to 0$ (resp., $+\infty$), then less (resp., more) subsets are likely to be included in $\mathcal{C}$, and thus the class tree is close to a star (resp., a binary tree). Hence it is expected that a larger δ decomposes the given N-class problem into more subproblems for fewer classes.
- It is possible that $\mathcal{C}$ contains more than one maximal subset, i.e., more than one class tree. In such a case, we have to choose the class tree used for classifying a future example, but we omit the details due to space limitation. (In our preliminary experiments, we hardly observed such a case.)

3.2 Experimental Results

We compare SS-TREE with other classifiers, C4.5 [25], LibSVM [2] and MCSVM [4], in terms of error rate on test sets. All the classifiers have some tunable parameters. We try some combinations of parameter values for each classifier, and evaluate it by the smallest error rate.

Table 4: Error rates (%) of SS-TREE, C4.5, LibSVM and MCSVM

Data	N	SS-TREE			C4.5	LibSVM	MCSVM
		$K = 10$	20	30			
LENSES	3	20.33	20.00	20.17	16.66	21.66	**0.00**
IRIS	3	5.80	5.80	5.80	4.66	**3.99**	4.67
WINE	3	10.47	10.69	10.74	9.08	**9.05**	**9.05**
BALANCE	3	*13.55	*13.23	*13.13	20.16	**8.63**	9.17
CMC	3	*45.25	*45.10	*44.90	45.41	**44.05**	45.43
CAR	4	3.30	3.35	3.46	2.83	**0.34**	1.24
NURSERY	5	0.97	0.73	0.73	0.62	0.04	**0.03**
BRIDGES	6	42.71	41.05	40.42	39.09	**38.18**	39.27
DERMATOLOGY	6	*14.33	*14.27	*14.32	14.99	12.53	**12.50**
ANNEAL	6	7.90	7.80	7.90	6.00	6.00	**4.00**
SAT	6	*15.71	*15.03	*15.01	16.70	**12.15**	12.40
ZOO	7	***0.00**	***0.00**	***0.00**	1.00	1.00	**0.00**
GLASS	7	*29.47	*28.28	*27.37	32.72	**25.17**	25.36
YEAST	10	*41.48	*40.98	*40.64	41.91	**40.22**	42.50
SOYBEAN	19	*10.77	*9.23	*9.76	12.76	**7.44**	9.19
AUDIOLOGY	24	*15.00	*12.69	***11.92**	15.38	34.61	23.08

SS-TREE: We set δ to ε, 1.0, 1.1, 1.2, 1.5, 2.0, 3.0 and $+\infty$, where ε is a sufficiently small positive number. We use DECISIONTABLE package of Weka [28] to generate a set $\mathcal{T}$ of decision tables. We set $K = |\mathcal{T}|$ to 10, 20 and 30.

C4.5: We test 8 combinations of parameter values: we set `confidence rate` to 1%, 25% (default), 50% or 99%, `binary split` option to `true` or `false` (default), and the other parameters to the default values.

LibSVM: We test 16 combinations: we use 2-class C-SVM and RBF kernel, and set $C = 0.5$, 1.0 (default), 2.0 or 4.0, $\gamma = 0.0$ (default), 0.5, 1.0 or 2.0, and the other parameters to the default values. Note that LibSVM employs one-to-one framework to extend binary C-SVM to multiclass one.

MCSVM: As it is an extension of binary C-SVM, LibSVM and MCSVM have similar parameters in common. For MCSVM, we test the same 16 combinations as LibSVM.

We show the results in Table 4. Boldface for each data set shows the smallest (i.e., best) error rate among all classifiers. A sign * on SS-TREE indicates that the error rate is smaller than C4.5. The effectiveness of SS-TREE is outstanding when N is large; for $N \geq 7$, SS-TREE outperforms C4.5 for all data sets and becomes more competitive with SVMs. In particular, SS-TREE is much better than the other classifiers for AUDIOLOGY, which has the largest N among the used data sets. For larger N, we may have to decompose N-class problem more carefully. The experimental results indicate that SS-TREE succeeds in extracting class trees which are effective in decreasing error rates.

4 Future Work

In this article, we have reviewed our visual classifier constructed by edge crossing minimization on bipartite graph. Our main future work is summarized as follows.

- We have assumed that a set $\mathcal{T}$ of decision tables is given and generated it by Weka in the experiments. We need to develop a faster algorithm to generate an effective set $\mathcal{T}$ of decision tables.
- In the experiments, we used a binarization algorithm to deal with a data set with numerical attributes since our formulation is limited to nominal data sets. We should consider an extended formulation that can treat numerical attributes directly.
- We also have to find application areas where our visual classifier is effective for data analysis and knowledge discovery.
- We need to work on the most essential question: Does edge crossing minimization on graphs really leads to success of learning?

References

[1] A. Asuncion and D.J. Newman. *UCI Machine Learning Repository.* University of California, Irvine, School of Information and Computer Sciences, 2007. `http://www.ics.uci.edu/~mlearn/MLRepository.html` (accessed on Nov. 30th, 2010).

[2] C. C. Chang and C. J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm` (accessed on Nov. 30th, 2010).

[3] L. Cheng, J. Zhang, J. Yang, and J. Ma. An improved hierarchical multi-class support vector machine with binary tree architecture. *International Conference on Internet Computing in Science and Engineering*, pp. 106–109, 2008.

[4] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, Vol. 2, pp. 265–292, 2001.

[5] T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, Vol. 2, pp. 263–286, 1995.

[6] P. Eades and N. C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, Vol. 11, pp. 379–403, 1994.

[7] J. H. Friedman. Recent advances in predictive (machine) learning. *Journal of Classification*, Vol. 23, pp. 175–197, 2006.

[8] M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, Vol. 4, pp. 312–316, 1983.

[9] K. Haraguchi, S.H. Hong, and H. Nagamochi. Bipartite graph representation of multiple decision table classifiers. In *Proc. SAGA 2009*, Vol. 5792 of *LNCS*, pp. 46–60. Springer, 2009.

[10] K. Haraguchi, S.H. Hong, and H. Nagamochi. Classification by ordering data samples. *RIMS Kokyuroku*, Vol. 1644, pp. 20–34, 2009. ISSN 1880-2818.

[11] K. Haraguchi, S.H. Hong, and H. Nagamochi. Classification via visualization of sample-feature bipartite graphs. Technical Report 2009-011, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Japan, 2009.

[12] K. Haraguchi, S.H. Hong, and H. Nagamochi. Visualization can improve multiple decision table classifiers. In *Proc. MDAI 2009 (ISBN: 978-84-00-08851-4)*, pp. 41–52, 2009.

[13] K. Haraguchi, S.H. Hong, and H. Nagamochi. Effectiveness of sample poset based visual classifier for data sets conceptualized by the number of attributes. In *Proc. WAAC 2010*, pp. 26–33, 2010.

[14] K. Haraguchi, S.H. Hong, and H. Nagamochi. Multiclass visual classifier based on bipartite graph representation of decision tables. In *Proc. LION4*, Vol. 6073 of *LNCS*, pp. 169–183. Springer, 2010.

[15] K. Haraguchi and H. Nagamochi. Extension of ICF classifiers to real world data sets. In *Proc. 20th IEA/AIE*, Vol. 4570 of *LNAI*, pp. 776–785. Springer, 2007.

[16] T. Hastie and R. Tibshirani. Classification by pairwise coupling. In *Advances in Neural Information Processing Systems*, Vol. 10. MIT Press, 1998.

[17] W. Huang, S.H. Hong, and P. Eades. Layout effects on sociogram perception. In *Graph Drawing*, Vol. 3843 of *LNCS*, pp. 262–273. Springer, 2006.

[18] F. Janssen and J. Fürnkranz. On the quest for optimal rule learning heuristics. *Machine Learning*, Vol. 78, pp. 343–379, 2010.

[19] M. Jünger and P. Mutzel. 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms. *Journal of Graph Algorithms and Applications*, Vol. 1, No. 1, pp. 1–25, 1997.

[20] R. Kohavi. The power of decision tables. In *ECML*, Vol. 912 of *LNAI*, pp. 174–189. Springer, 1995.

[21] S. Kumar, J. Ghosh, and M.M. Crawford. Hierarchical fusion of multiple classifiers for hyperspectral data analysis. *Pattern Analysis and Applications*, Vol. 5, No. 2, pp. 210–220, 2002.

[22] H. Nagamochi. An improved bound on the one-sided minimum crossing number in two-layered drawings. *Discrete and Computational Geometry*, Vol. 33, No. 4, pp. 569–591, 2005.

[23] N. J. Nilsson. *Learning machines.* McGraw-Hill, 1965.

[24] H. Purchase. Which aesthetic has the greatest effect on human understanding? In *Graph Drawing*, Vol. 1353 of *LNCS*, pp. 248–261. Springer, 1997.

[25] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[26] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-11, No. 2, pp. 109–125, 1981.

[27] S. M. Weiss and C. A. Kulikowski. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems.* Morgan Kaufmann, 1991.

[28] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, San Francisco, 2nd edition, 2005. `http://www.cs.waikato.ac.nz/ml/weka/` (accessed on Nov. 30th, 2010).